Problem Set 1

The following problem set will be worth 100 points. The code will be submitted electronically via Canvas using the "Problem Set 1" dropbox. The assignment is **due at the start of the class two weeks from the date it was assigned**.

Your code will be graded on both elegance and *user-friendliness*.

Exercise #1 – Fraction Problem (10pts)

Define a class called Fraction. This class is used to represent a ratio of two integers. Include mutator methods which allow users to set values for the numerator and the denominator. Also include a method that returns the value of the numerator divided by the value of the denominator as a double. Include an additional method that outputs the value of the fraction reduced to lowest terms (e.g. instead of outputting 20/60, output 1/3.) This will require finding the greatest common divisor for both the numerator and denominator, and then dividing both by that number. Include testing examples in the main method of your class.

Save your solution in the file named **Fraction.java**.

Exercise #2 – Driving Problem (10pts)

Create a Java class named `Odometer` which can be used to calculate fuel and mileage for an automobile. The class should have instance data which tracks the miles driven and the fuel efficiency in miles per gallon. Include a mutator method to reset the odometer to zero miles, a mutator method to set the fuel efficiency, a mutator method which accepts the miles driven for a trip and adds it to the odometer's total, and an accessor method that returns the number of gallons of gasoline that the vehicle has consumed since the last reset. Include a main method that creates an instance of the class and invokes the methods for testing.

Save your solution in a file named **Odometer.java**.

Exercise #3 – Seating Problem (20pts)

Assume an airplane seats passengers with seat numberings as follows:

1	A	B	C	D
2	A	B	C	D
3	A	B	C	D

4	A	B	C	D
5	A	B	C	D
6	A	B	C	D
7	A	B	C	D

Write a class named **Airplane** which holds this seating arrangement in some sort of data structure (two dimensional array or Java collection class or similar.) Write a `main` method in this class which will loop, allowing the user to enter a row number and column letter for each seat occupied until no more seats need to be added. Print out the grid after each pass of the loop, marking each assigned seat with an **X**. Show the user an error message if the user attempts to set an already assigned seat.

Save your solution in a file named **Airplane.java**.

Exercise #4 – Queueing Problem (20pts)

In data structures, a *queue* is a data structure that is "first in, first out". It contains two methods:

- enqueue – adds an item to the queue
- dequeue – removes an item from the queue

For this assignment you are going to implement a variation on this called a **priority queue**, where an integer corresponding to the priority is passed into the *enqueue* method upon invocation. As an example, we can have three successive calls to add items to the queue:

- q.enqueue("X", 10)
- q.enqueue("Y", 1)
- q.enqueue("Z", 3)

Invoking *dequeue* on the object will then remove the item "X" from the queue, since it has the highest priority. A second invocation of the method returns "Z" while the third invocation returns "Y". Use an **ArrayList** as a representation of the queue. Include a main method which demonstrates successive adds and removals from your queue.

Save your solution in a file named **PriorityQueue.java**.

Exercise #5 – Pizza Problem (15pts)

Create a class named Pizza that stores information about a single pizza. It should contain the following:

- private instance variables to store the size of the pizza (small, medium, large), number of cheese toppings, number of pepperoni toppings and the number of ham toppings
- Constructor(s) that set all instance variables

- Public methods to get/set the instance variables
- A public method named *calcCost()* that returns a double that is the cost of the pizza. Pizza cost is determined by:
    - Small: $10 + $2 per topping
    - Medium: $12 + $2 per topping
    - Large: $14 + $2 per topping
- A public method named *getDescription*() that returns a string containing the pizza size, quantity of each topping and the pizza cost as calculated by *calcCost*().

Write test code to create several pizzas and their descriptions in the main method of the class. For instance a large pizza with one cheese, one pepperoni and two ham toppings should cost $22. Name your class **Pizza.java**.

Exercise #6 – Number List Problem (15pts)

Write a program that reads numbers from the keyboard into an array of type **int[]**. You can assume that there won't be more than 50 elements added to the list. Your output should be a two column list. The first is the set of distinct array elements, the second column is the number of occurrences of each element. The list should be sorted on entries in the first column, largest to smallest. As an example for the array itself:

{-12, 3, -12, 4, 1, 1, -12, 1, -1, 1, 2, 3, 4, 2, 3, -12}

Your output should look like the following:

N	Count
4	2
3	3
2	2
1	4
-1	1
-12	4

Save your solution in a file named **IntList.java**.

Exercise #7 – Doctor Problem (10pts)

Give the definition of a class named **Doctor** whose objects are records for a clinic's doctors. This class will be a derived class of the class **SalariedEmployee**. A salaried employee consists of:

- First Name (String)

- Last Name (String)
- Hire Date (String)
- Salary (double)

A **Doctor** record has the doctor's specialty (such as "Pediatrician", "Obstetrician", "General Practitioner", and so forth; so use the type **String**) and office visit fee (use type **double**.) Be sure your class has a resolvable complement of constructors, accessor and mutator methods, and suitably defined **equals** and **toString** methods. Write a program to test all of your methods.

Submission Requirements: Submit the aforementioned files in a zip file with the naming strategy:

First initial + last name + PS + problem set number.zip

As an example, I would submit the code in a zip file named **mmeluskyPS1.zip**. Submit your zip file via the "Problem Set 1" Canvas dropbox before the date of the close of the assignment.